

An initial simple example looks like the following:

```

MPI_File fh;
MPI_Status status;
/* MPI_Status is a structure containing:
   MPI_SOURCE - id of processor sending the message
   MPI_TAG - the message tag
   MPI_ERROR - error status */
MPI_Init(&argc,&argv);
MPI_Comm_rank(MPI_COMM_WORLD, &rank);
MPI_Comm_size(MPI_COMM_WORLD, &nprocs);

bufsize = FILESIZE/nprocs;
nints = bufsize/sizeof(int);
int *buf = (int *)malloc(bufsize);

MPI_File_open(MPI_COMM_WORLD, "datafile", MPI_MODE_RDONLY, MPI_INFO_NULL,
&fh);
/*      File modes given to MPI_File_open() can be the combination of

      MPI_MODE_RDONLY          open for read only
      MPI_MODE_RDWR             open for reading and writing
      MPI_MODE_WRONLY           open for write only
      MPI_MODE_CREATE           create the file if it does not exist
      MPI_MODE_EXCL             generate an error if creating a file that already exists
      MPI_MODE_DELETE_ON_CLOSE delete the file on MPI_File_close is called
      MPI_MODE_APPEND           set initial position of all file pointers to end of file

      combined in C with the bitwise or operator, in fortran combined by addition.      */

MPI_File_seek(fh, rank * bufsize, MPI_SEEK_SET);
/*      MPI_SEEK_SET: the pointer is set to offset
      MPI_SEEK_CUR: the pointer is set to the current pointer position plus offset
      MPI_SEEK_END: the pointer is set to the end of file plus offset      */

MPI_File_read(fh, buf, nints, MPI_INT, &status);
MPI_Barrier(MPI_COMM_WORLD);
MPI_File_close(&fh);

```

This will read a single file splitting up the contents across all the MPI ranks.
Very posix like but just done in parallel.

To do the same thing but without the explicit seek call:

```

MPI_OFFSET_KIND offset;
nints = FILESIZE / (nprocs*sizeof(int));
offset = rank * nints * sizeof(int);
MPI_FILE_READ_AT(fh, offset, buf, nints,MPI_INT, &status);

```

Or yet another way using the MPI_File_set_view() routine:

```

MPI_File thefile;

for (i=0; i<BUFSIZE; i++)
    buf[i] = rank * BUFSIZE + i;

MPI_File_open(MPI_COMM_WORLD, "testfile", MPI_MODE_CREATE |
MPI_MODE_WRONLY,
              MPI_INFO_NULL, &thefile);
MPI_File_set_view(thefile, rank * BUFSIZE * sizeof(int), MPI_INT, MPI_INT, "native",
                  MPI_INFO_NULL);
MPI_File_write(thefile, buf, BUFSIZE, MPI_INT, MPI_STATUS_IGNORE);
MPI_File_close(&thefile);

```

But a more useful and complex example of set_view is as follows:

```

MPI_Aint lb, extent;
MPI_Datatype etype, filetype, contig;
MPI_Offset disp;

/* create a data type that is 1 floating point */
MPI_Type_contiguous(1, MPI_FLOAT, &contig);
MPI_Type_commit(&contig);

/* the total (in file) data block is 3 of these new data types without any padding */
lb = 0; extent = 3 * sizeof(float);

```

```
/* create the new type and commit it
MPI_Type_create_resized(contig, lb, extent, &filetype);
MPI_Type_commit(&filetype);

/* the input file has 5 integers in the beginning we skip */
npts = total_points / nprocs;
disp = 5 * sizeof(int) + rank * npts * sizeof(float);

MPI_File_open(MPI_COMM_WORLD, "datafile", MPI_MODE_RDWR, MPI_INFO_NULL, &fh);
MPI_File_set_view(fh, disp, contiq, filetype, "native", MPI_INFO_NULL);
MPI_File_read(fh, buf, npts, contiq, MPI_STATUS_IGNORE);
```

What does this code section actually do:

the file contains x/y/z for each node point, the above will just read all the x values out of the file.